

# Object-Oriented Programming Using C++

---

Ira Pohl



**The Benjamin/Cummings Publishing Company, Inc.**

Redwood City, California • Menlo Park, California

Reading, Massachusetts • New York • Don Mills, Ontario • Wokingham, U.K.

Amsterdam • Bonn • Sydney • Singapore • Tokyo • Madrid • San Juan

# CONTENTS

---

## 1

### WHY OBJECT-ORIENTED PROGRAMMING IN C++? 2

- 1.1 OBJECT-ORIENTED PROGRAMMING 4
- 1.2 WHY C++ IS A BETTER C 6
- 1.3 ENCAPSULATION AND TYPE EXTENSIBILITY 8
- 1.4 CONSTRUCTION OF OBJECTS 9
- 1.5 CONVERSIONS, OPERATORS, AND SEAMLESS TYPES 12
- 1.6 INHERITANCE 13
- 1.7 POLYMORPHISM 16
- 1.8 BENEFITS OF OBJECT-ORIENTED PROGRAMMING 19
- 1.9 REFERENCES 20

## 2

### NATIVE TYPES AND STATEMENTS 22

- 2.1 PROGRAM ELEMENTS 24
  - Comments 24
  - Keywords 24
  - Identifiers 25
  - Literals 25
  - Operators and Punctuators 26
- 2.2 INPUT/OUTPUT 27
- 2.3 PROGRAM STRUCTURE 28

- 2.4 SIMPLE TYPES 30
  - Initialization 31
- 2.5 THE TRADITIONAL CONVERSIONS 33
- 2.6 ENUMERATION TYPES 36
- 2.7 EXPRESSIONS 37
- 2.8 STATEMENTS 42
  - Assignments and Expressions 42
  - Compound Statements 43
  - The *if* and *if-else* Statements 44
  - The *while* Statement 45
  - The *for* Statement 45
  - The *do* Statement 47
  - Transfer Statements 47
  - The *break* and *continue* Statements 48
  - The *switch* Statement 49
  - The *goto* Statement 50
- 2.9 SUMMARY 51
- 2.10 EXERCISES 53

### 3

## FUNCTIONS AND POINTERS 60

- 3.1 FUNCTIONS 61
  - Function Invocation 62
- 3.2 FUNCTION DEFINITION 62
- 3.3 THE *return* Statement 64
- 3.4 FUNCTION PROTOTYPES 65
- 3.5 DEFAULT ARGUMENTS 67
- 3.6 OVERLOADING FUNCTIONS 67
- 3.7 INLINING 69
- 3.8 SCOPE 70
- 3.9 STORAGE CLASS 71
  - The Storage Class *auto* 71
  - The Storage Class *register* 72
  - The Storage Class *extern* 73
  - The Storage Class *static* 74
  - Linkage Mysteries 76
- 3.10 POINTER TYPES 76
  - Addressing and Dereferencing 77
  - Simulating Call-by-Reference 78

- 3.11 REFERENCE DECLARATIONS AND CALL-BY-REFERENCE 79
- 3.12 THE USES OF *void* 81
- 3.13 ARRAYS AND POINTERS 83
  - Subscripting 84
  - Initialization 85
- 3.14 THE RELATIONSHIP BETWEEN ARRAYS AND POINTERS 85
- 3.15 PASSING ARRAYS TO FUNCTIONS 87
- 3.16 STRINGS: A KERNEL LANGUAGE ADT 89
- 3.17 MULTIDIMENSIONAL ARRAYS 90
- 3.18 FREE STORAGE OPERATORS *new* AND *delete* 91
- 3.19 SUMMARY 94
- 3.20 EXERCISES 96

## 4

### IMPLEMENTING ADTs IN THE BASE LANGUAGE 104

- 4.1 THE AGGREGATE TYPE *struct* 105
- 4.2 STRUCTURE POINTER OPERATOR 107
  - typedef* and Casting 108
- 4.3 AN EXAMPLE: STACK 109
- 4.4 UNIONS 114
- 4.5 COMPLEX NUMBERS 117
- 4.6 AN EXAMPLE: FLUSHING 119
- 4.7 BIT FIELDS 125
- 4.8 AN EXAMPLE: DYNAMIC ARRAYS 127
- 4.9 SUMMARY 130
- 4.10 EXERCISES 131

## 5

### DATA HIDING AND MEMBER FUNCTIONS 136

- 5.1 MEMBER FUNCTIONS 139
- 5.2 VISIBILITY *private* AND *public* 142
- 5.3 CLASSES 144
- 5.4 CLASS SCOPE 145
  - Scope Resolution Operator `::` 146
  - Nested Classes 147

- 5.5 *static* MEMBER 147
- 5.6 AN EXAMPLE: REVISITING FLUSHING 148
- 5.7 THE *this* POINTER 152
- 5.8 *static* AND *const* MEMBER FUNCTIONS 153
- 5.9 CONTAINERS AND ITEMS ACCESS 155
- 5.10 SUMMARY 158
- 5.11 EXERCISES 159

## 6

### OBJECT CREATION 166

- 6.1 CLASSES WITH CONSTRUCTORS 168
  - The Default Constructor 170
- 6.2 CONSTRUCTING A DYNAMICALLY SIZED STACK 171
  - The Copy Constructor 172
  - Constructor Initializer 173
- 6.3 CLASSES WITH DESTRUCTORS 174
- 6.4 AN EXAMPLE: DYNAMICALLY ALLOCATED STRINGS 175
- 6.5 A CLASS *vect* 179
- 6.6 MEMBERS THAT ARE CLASS TYPES 181
- 6.7 AN EXAMPLE: A SINGLY LINKED LIST 183
- 6.8 POLYNOMIALS AS A LINKED LIST 188
- 6.9 STRINGS USING REFERENCE SEMANTICS 193
- 6.10 NO CONSTRUCTOR, COPY CONSTRUCTOR, AND OTHER MYSTERIES 196
  - new*, *delete* Options and Syntax 197
  - Destructor Details 198
  - Constructors as Conversions 199
- 6.11 SUMMARY 200
- 6.12 EXERCISES 202

## 7

### AD HOC POLYMORPHISM 210

- 7.1 CLASS-DEFINED CONVERSIONS 212
- 7.2 OVERLOADING AND FUNCTION SELECTION 214
- 7.3 *friend* FUNCTIONS 217
- 7.4 OVERLOADING OPERATORS 220
- 7.5 UNARY OPERATOR OVERLOADING 221

7.6	BINARY OPERATOR OVERLOADING	224
7.7	OVERLOADING ASSIGNMENT AND SUBSCRIPTING OPERATORS	226
7.8	MORE SIGNATURE MATCHING	230
7.9	POLYNOMIAL: TYPE AND LANGUAGE EXPECTATIONS	233
7.10	SUMMARY	235
7.11	EXERCISES	237

## 8

### VISITATION: ITERATORS AND CONTAINERS 242

8.1	VISITATION	243
8.2	ITERATORS	246
8.3	AN EXAMPLE: <i>quicksort</i> ()	247
8.4	FRIENDLY CLASSES AND ITERATORS	253
8.5	OVERLOADING OPERATOR () FOR INDEXING	256
8.6	OVERLOADING <i>new</i> AND <i>delete</i>	259
8.7	POINTER OPERATORS AND SMART POINTERS	262
	Pointer to Class Member	264
8.8	GENERICITY WITH <i>void*</i>	267
8.9	SUMMARY	269
8.10	EXERCISES	270

## 9

### INHERITANCE: SUBTYPING AND CODE REUSE 274

9.1	A DERIVED CLASS	276
9.2	TYPING CONVERSIONS AND VISIBILITY	278
9.3	CODE REUSE: A DYNAMIC ARRAY BOUNDS	281
9.4	CODE REUSE: A BINARY TREE CLASS	283
9.5	VIRTUAL FUNCTIONS	288
9.6	ABSTRACT BASE CLASSES	292
9.7	MULTIPLE INHERITANCE	298
9.8	INHERITANCE AND DESIGN	303
	Subtyping Form	304
9.9	DETAILED C++ CONSIDERATIONS	306
9.10	SUMMARY	307
9.11	EXERCISES	309

## 10

<b>PARAMETRIC POLYMORPHISM</b>	<b>316</b>
10.1	TEMPLATE CLASS STACK 319
10.2	FUNCTION TEMPLATES 321
	Signature Matching and Overloading 322
10.3	CLASS TEMPLATES 324
	Friends 324
	Static Members 325
	Class Template Arguments 325
10.4	PARAMETERIZING THE CLASS <i>vect</i> 327
10.5	PARAMETERIZING <i>quicksort</i> () 331
10.6	PARAMETERIZED BINARY SEARCH TREE 336
10.7	INHERITANCE 339
10.8	OWNERSHIP AND DESIGN ISSUES 341
10.9	DETAILED CONSIDERATIONS 342
10.10	SUMMARY 343
10.11	EXERCISES 345

## 11

<b>EXCEPTIONS</b>	<b>348</b>
11.1	USING <i>assert.h</i> 350
11.2	USING <i>signal.h</i> 351
11.3	C++ EXCEPTIONS 356
11.4	THROWING EXCEPTIONS 357
11.5	<i>try</i> BLOCKS 360
11.6	HANDLERS 361
11.7	EXCEPTION SPECIFICATION 362
11.8	<i>terminate</i> () AND <i>unexpected</i> () 363
11.9	EXAMPLE EXCEPTION CODE 363
11.10	THE PHILOSOPHY OF ERROR RECOVERY 366
11.11	SUMMARY 367
11.12	EXERCISES 370

## 12

<b>OOP USING C++</b>	<b>374</b>
12.1	OOP LANGUAGE REQUIREMENTS 375
12.2	ADTs IN NON-OOP LANGUAGES 377

12.3	CLIENTS AND MANUFACTURERS	378
12.4	REUSE AND INHERITANCE	379
12.5	POLYMORPHISM	380
12.6	LANGUAGE COMPLEXITY	381
12.7	C++ OOP BANDWAGON	383
12.8	PLATONISM: TABULA RASA DESIGN	384
12.9	DESIGN PRINCIPLES	386
12.10	SCHEMA, DIAGRAMS, AND TOOLS	387
12.11	LAST WORDS	389
12.12	REFERENCES	391
12.13	SUMMARY	392
12.14	EXERCISES	394

## APPENDIX A: ASCII CHARACTER CODES 396

## APPENDIX B: OPERATOR PRECEDENCE AND ASSOCIATIVITY 398

## APPENDIX C: C++ LANGUAGE GUIDE 400

C.1	LEXICAL ELEMENTS	402
	Comments	403
	Identifiers	403
	Keywords	403
C.2	CONSTANTS	404
C.3	DECLARATIONS AND SCOPE RULES	406
C.4	LINKAGE RULES	409
C.5	TYPES	411
C.6	CONVERSION RULES	413
C.7	EXPRESSIONS AND OPERATORS	415
	<i>size of Expressions</i>	415
	Autoincrement and Autodecrement Expressions	416
	Arithmetic Expressions	416
	Relational, Equality, and Logical Expressions	417
	Assignment Expressions	418
	Comma Expressions	419
	Conditional Expressions	419
	Bit Manipulation Expressions	420
	Address and Indirection Expressions	420

	<i>new</i> and <i>delete</i> Expressions	421
	Placement and Syntax Overloading	422
	Other Expressions	424
C.8	STATEMENTS	424
	Expression Statements	425
	The Compound Statement	425
	The <i>if</i> and The <i>if-else</i> Statements	425
	The <i>while</i> Statement	426
	The <i>for</i> Statement	427
	The <i>do</i> Statement	428
	Transfer Statements	428
	The <i>break</i> and <i>continue</i> Statements	428
	The <i>switch</i> Statement	430
	The <i>goto</i> Statement	431
	The Declaration Statement	431
C.9	CLASSES	432
	Constructors and Destructors	433
	Member Functions	434
	The <i>this</i> Pointer	434
	<i>static</i> and <i>const</i> Member Functions	435
	Inheritance	437
	Multiple Inheritance	438
	Constructor Invocation	440
	Abstract Base Classes	442
	Pointer to Class Member	442
C.10	FUNCTIONS	442
	Prototypes	443
	Overloading	443
	Call-by-Reference	446
	Inline	447
	Default Arguments	447
	Friend Functions	448
	Operator Overloading	449
	Virtual Functions	450
	Type-Safe Linkage	452
C.11	TEMPLATES	452
	Function Template	453
	Friends	455
	Static Members	455

C.12	EXCEPTIONS	455
	Throwing Exceptions	456
	Try Blocks	457
	Handlers	458
	Exception Specification	459
	<i>terminate ()</i> and <i>unexpected ()</i>	459
C.13	CAUTION AND COMPATIBILITY	460
	Nested Class Declaration	460
	Type Compatibilities	460
	Miscellaneous	461
	Unimplemented Features	461
C.14	STYLE EXAMPLES AND PRAGMATICS	462
	Class Definition Style	463

## APPENDIX D. INPUT/OUTPUT 466

D.1	THE OUTPUT CLASS <i>ostream</i>	468
D.2	FORMATTED OUTPUT AND <i>iomanip.h</i>	469
D.3	USER-DEFINED TYPES: OUTPUT	471
D.4	THE INPUT CLASS <i>istream</i>	473
D.5	FILES USING <i>fstream.h</i>	475
D.6	THE FUNCTIONS AND MACROS IN <i>cctype.h</i>	479
D.7	USING THE STREAM STATES	480
D.8	MIXING I/O LIBRARIES	483

## INDEX 485